Security Without Firewalls

A tutorial

Abe Singer

Copyright 2005 Abe Singer

The Big Picture

Copyright 2005 Abe Singer

The Myth of Firewalls

Crunchy outside, soft chewy inside

Users are not to be trusted

You have to secure the hosts anyways

Copyright 2005 Abe Singer

There's a pervasive myth that firewalls are necessary for security.

Much literature, especially in trade rags, recommend using a firewall as the number one thing for security

Visa Cardholder Information Security program require one

But they usually don't tell you how they should be configured

Many people think you're doing something wrong if you don't have one

But firewalls are just things that let some things through and not others

They work only as well as they are configured

They're vulnerable to cascade failures – multiple dependencies

Have performance problems

Only work if traffic flows through them, often have problems with asymmetric routes

You have to worry about your users anyways, because if a user account is compromised, or the user is misbehaving, that's the point where you are screwed

Goals

Protect, Detect, Respond

Confidentiality, Integrity, Availability

It's all about the data

Copyright 2005 Abe Singer

Your goals in configuring a secure infrastructure…

A common process for managing security is:
Protect what you can, Detect what you can't protect, Respond to what you can detect
---
But what are you trying to protect?

one view of the attributes that you want to protect/provide are:
Confidentiality, Integrity, Availability (CIA)
---
but what does that actually mean? Ultimately, it means protecting data.

What data?

Operating system

Applications

User data

Everything!

Copyright 2005 Abe Singer

It usually comes down to protecting data, whether it be OS data such as binaries which can be used to gain privileges, or user data which is irreplaceable

Strategy

Building an architecture

Configuration Management

No Plaintext Passwords

Patching

Monitor and maintain

Copyright 2005 Abe Singer

And while I'm going to present a combination of techniques and tactics, it's not just about some tricks, but building and maintaining an architecture

---

The core strategy we (SDSC) follows, to achieve the goals, are

Configuration managements – being able to maintain configs across patches and upgrades, and ensuring all the hosts are configured the way we think they are (assurance). This includes both OS configuration and services provided

Configuration management has two parts: A reference system, which is a known good host configuration, and use of configuration management to maintain consistency of configs and software across hosts

---

No plaintext passwords, because attackers love to sniff networks

---

Patching, because almost everyone, including SDSC, gets owned as the result of failing to get patches applied properly

---

And once you set machines up, you can't just forget about them. you need to monitor/verify that things are running the way they should be. You need to do this anyways to detect hardware/system failures.

And you want to be able to do things consistently across systems

Planning

What are you running?

How do you protect it?

What's your network look like?

What should it look like?

Copyright 2005 Abe Singer

Things to do to get started on your architecture

First, identify what services are you are running – type of service, not the particular technology

---

What kind of servers? DHCP? DNS? NTP? HTTP? NFS? Samba? LDAP? etc.

Can the service protect itself (through proper configuration?)
Does it require authentication? If so, is it strong authentication? (Will explain later)
Does it require privileged access? Run as root, or setuid?
What sort of access control do you need for your services?

If it can't protect itself, can you protect it through other means?
Or do you really need to run it?

What's your current network design? Do you know? does anyone? this may sound silly, but a lot of places don't have a good picture of how the network is configured. Or the picture only exists inside the head of the network manager. And then sometimes reality doesn't quite match the picture. Bill Cheswick can tell stories of mapping networks only to have an exec say "that line was supposed to be disconnected a year ago!"

So, regardless of what your network looks like, what *should* it look like?

Can you separate "trusted" or "managed" hosts from untrusted?
What are the trust relationships between hosts?

Trust Relationships

"I do not think that word means what you think it means."

-- Inigo Montoya

Copyright 2005 Abe Singer

Trust relationships are a term often used but not often understood. Trust relationships between machines are often exploited as a path to gain access to different hosts

Trust is the expectation that a person or thing will behave as expected. In machine terms, trust is a level of confidence that a system will behave as expected or that information provided is accurate

A trust relationship is when two machines interacting have the expectation that the other machine is providing accurate information

Trust

NFS

DNS

HTTPS

UDP packets (IP addresses for that matter)

Syslog

Allowing remote root access

DHCP

NIS

Remote user login

Copyright 2005 Abe Singer

Examples:

NFS- server expects UID from client to be accurate, and doesn't validate file handles once generated. Once you have a filehandle, if you can get packets to/from the NFS serrver you can modify the file

DNS -Root servers reporting accurate NS record, for starters. Then there's reverse lookups -- there's nothing to stop a name server for providing any domain on a reverse lookup -- forward and reverse are not automagically synchronized.

HTTPS – server is who they say they are which is why we have certs and CRLs. (and How many people actually update their CRLs, hmmmm?)

UDP packets (IP addresses for that matter)

Syslog -- accepts from any host, doesn't authenticate host in any way

DHCP -- NIS

Remote access

Transitive trust -- you assume that remote host is secure and has authenticated user, and that they are the only one who could supply the appropriate credentials (e.g. userrname, password).

Allowing remote root access

A specifically scary issue of transitive trust, you have to believe that nobody can intercept the root password on the remote host.

Remote user login – if they have the right username and password, must be them

Network trust

control of hosts

spoofing

sniffing

Copyright 2005 Abe Singer

Some services require a certain amount of trust of the network on which it is running

NFS requires a *lot*

tcp-wrappers require some

ssh doesn't *require* any, but can if you do IP-based access controls without using hostbased authentication.

But what it comes down to in your architecture, are you in control of what hosts are allowed on a given network

---

If you can't do that, you're not very safe from address spoofing, man-in-the-middle-attacks, etc.

And if your protocols use plaintext, they are vulnerable to sniffing

---

AKA interception of traffic.

And if your service is vulnerable to network-based attacks, and you can't control what's on the network, you shouldn't run the service. Or you shouldn't send the data over the cleartext channel, etc.

Yeah, I know, try telling that to the VP of marketing who's promised to roll something out, but that's the point here. And put it back to the CIO with "and what happens when one of our machines starts attacking the server?"

Spoofing

IP address

IP MITM

ARP (MITM)

DNS

Copyright 2005 Abe Singer

A bit on types of spoofing attacks, just so we all know what we're talking about.

Therre are a variety, each of which takes a different type of access to the network or data

---

The simplest is IP address spoofing. This is often used in distributed denial of service attacks, to hide the origin of the attacks. Any host can send a packet to a target using any source address, as most network do not bother to validate the source address of outbound packets.

Usually this is used for one-way communication, as any replies from the target will be routed back to the source address, not the host who is forging the address (how would anyone know where to route it?) So this type of spoofing is not seen on TCP-based connections, unless

the attacker has control of something between the two endpoints, in which case they can do

an IP-based Man-in-the-middle attack

.....

These are pretty rare -- the attacker has to have control of at least one router between you and him(her)self, on/or some control over the routing of packets. Not impossible, but relatively unlikely. Nevertheless, doing IP-based address controls for address blocks that you don't control is impossible to assure.

IP-based access controls should work for your network, if you've implemented anti-spoof filters on your border routers. You have, haven't you?

Another type of attack that has to be done on a local network is arp-spoofig

---

Also a type of MITM attack. This requires a host to be able to send ARP messages to the two hosts, claiming to have the IP address of the other. Each host then send its packets to the MAC address of the attacker, who accepts them and masquerades as the other. Since ARP normally only works on a local network segment, this requires an attacker to have acces to a host on the network segment, and to be able to forge packets, which normally requires root access.
Which is why a lot of this talk is about controlling root access.

Network Design

Service that are vulnerable

Services that aren't

Copyright 2005 Abe Singer

So, knowing what you know about networks and spoofing, here's the simple network design strategy:

For services that are vulnerable to spoofing,

---

only allow them to talk to networks that you control

Services that aren't,

---

well, maybe they can talk to anything, but do they really need to?

"Best block, not be there"

- Mr. Miyage

Copyright 2005 Abe Singer

This quote from the Karate Kid (courtesy of Bill Cheswick), says it all…

Why have things reachable if they don't need to be? Rather than have vulnerable services and hope to block them from evil hosts, don't run the service!

Reference System Overview

Known good config

Repeatable process

Centralized Stuff

Copyright 2005 Abe Singer

What is a referenece system?

A known good configuration, one that you've built yourself

An autoinstaller so that the config is repeatable

Install, patch, and run cfengine

Nightly configuration management

You'll want to centralize certain things, like user account management, home dirs, logging, etc.

Having home directories on a file server means that no data is lost when a system is upgraded and reinstalled.

Maintaining applications on a file server ensures consistency across hosts

Of course, if FS goes down, *everyone* goes down. But spend your money on reliable FS instead of having to keep *everything* up all the time

Configuration Management

Consistency

Self Healing

Knowledge Base

Copyright 2005 Abe Singer

Configuration management is a system where configuration of systems is centrally maintained and distributed to all hosts that need it.

CM is good for maintaining any file that diverges from the default OS installation, and those that are critical for system operation. E.g. inetd.conf, resolv.conf, sshd_config, hosts.allow. Plus binaries that you build and want installed locally, or binaries that you really really care about

---

As opposed to “standard configuration”, configuration management can allow for variations between systems, the point is to maintain a core consistency for critical stuff

And patches often rewrite those config files you spent hours figuring out, CM provides for a “self-healing” mechanism.

CM allows you to keep a record of changes, and by whom (with some help from revision control)

And allows you to leverage the knowledge of multiple sysadmins

---

-- you don’t lose the secret config file when they leave the organization

Privileged Access

Who's got root?

Sudo

No root logins

Copyright 2005 Abe Singer

Who gets root and why?

This is actually really important, being able to know who *could* have done something at a given time, plus knowing points of attack for intruders to gain privileges. If you don't know how has it, you're in trouble. Especially if they leave the organization...

Controlling access:

sudo

---

don't have to give root password, can limit actions, but not perfect

want to use something other than user password -- require two passwords to get privs, e.g. kerberized sudo

su

uid 0 accounts

Don't allow remote root logins

Know who and why

if you don't know who has it, change it and see who asks for it. People who get the root password should agree to not give it out to someone, that someone has to come to you to get it.

be sure to change root pw when someone leaves (and group membership). This means you need to be plugged into the HR loop so that you're always notified.

-revoke their privs if they misbehave

-don't give accounts on critical hosts unless they have privs and need access

* 9 times out of 10, if you ask them to explain why they need root, and why they can't do it some

Account Management

Consistency

Control

Copyright 2005 Abe Singer

You want centrally managed accounts

Concistency of uids and usernames
- for NFS or like
- for file management, such as tar, restore, etc.

- easier to track users across hosts
- easily add and remove accounts everywhere
  - want to be able to limit to class of hosts, or individual host

---

Issues:
- managing account/password file -- need to keep track of who goes where
- shadow file -- need to transport securely, and users need to be able to change pw somewhere
- Groups file too
- might want home dirs on file server, not local disk
- where does incoming email go?
- Where do their certs and kerberos principals go?

Methods:
- NIS
- LDAP
- Homegrown

Some ideas

No Plaintext Passwords

Copyright 2005 Abe Singer

No plaintext password

Opportunistic sniffing

Clue-challenged users

Strong authentication

Copyright 2005 Abe Singer

Script kiddies will often sniff passwords, even if that's not their primary intent

Any networks outside you control (assuming that your own network is under your control) is susceptible to sniffing

And even on your own network, it just makes sense -- it's much easier to sniff a network than to sniff a keyboard

---

Plus, users will try the insecure method if you give them a chance, or can't be bothered to change settings, etc.

The goal: passwords to do not get transmitted in a manner that allows for their interception

Can also be called strong authentication

Note that this doesn't protect you much when the attacker controls the endpoint

Getting there: a combination of tools, there's no one single solution

It takes a combination of things

Login auth,

Email,

web

No Plaintext Passwords

SSH and Kerberos

APOP, POP/SSL, IMAP/SSL

HTTPS (and Kerberos)

Postgres

AFS

NFS and IPSec

Copyright 2005 Abe Singer

Take a look at the various services you provide which authenticate things
Can it be limited to a trusted net?
Can it protect passwords itself -- encrypt, and exclude plaintext?
Are there clients the users can use?
or what clients are they using?
(It's okay to have multiple solutions)

If server doesn't support, is there an alternative? Can you ssl-wrap?

Login: ssh, kerberos (ssh supports kerberos)

Email: combination of things -- APOP, POP/SSL, IMAP/SSL
Can tunnel SSH to talk to an SMTP server

HTTPS for web pages, can use kerberos
Need a cert for the server

Postgres supports certs and kerberos, don't know about mysql or Oracle

AFS supports Kerberos. For NFS, you have to use something like IPSec

SSLwrap can be useful for things which don't support SSL themselves

IPSec is also a possibility for hosts that you manage, but you still have to support the others for things that are outside your network

SSLWrap

```
Inetd.conf:
   imap stream tcp nowait root /usr/local/etc/tcpd
     /usr/local/etc/imapd
    imaps stream tcp nowait nobody
     /usr/local/etc/tcpd /usr/local/etc/imapsd
Imapsd:
   /usr/local/etc/sslwrap -cert
     /usr/local/certs/ssl-imap.pem \ -CAfile
     /usr/local/certs/ca-cert.pem -port 143
tcpd:
   imapd: 127.0.0.1: allow
   imapd: ALL: rfc931: DENY
   imapsd : ALL : rfc931 : ALLOW
```

Copyright 2005 Abe Singer

Example of using sslwrap to encrypt imap connections.

Note that this could work just as well with a shared-key form of access, what do you really need PKI for in this context?

You could also tunnel over ssh, stunnel, etc. Pick whatever you like to work with

Building A Reference System

Known-good configuration

Fix what the vendor broke

Copyright 2005 Abe Singer

A reference system -- a “known good” host configuration that’s managed with cfengine
Hosts are allowed to deviate from the ref, not “standardized” systems
---
Start by installing the OS with vendor installer, try “everything” -- really!

Delete packages that:
start servers, install unnecessary setuid programs,
check for accounts that may have been created for a service

Examine all setuid/setgid programs, disable those that are unnecessary

Check directory and file ownership and permissions.
any world writeable?
stuff on root’s path

Note that there’s really no concern about things that dont’ affect root and don’t allow remote access. What does it matter if user have tools that just run in userspace? They can’t really use them to elevate privileges, so let them have whatever packages they want.

Use cfengine to make sure that any “fixes” stick across upgrades/patches
Change startup (/etc/rc*.d) to start up only desired services
Tcp-wrap appropriately
Reboot until it starts up the way you want it
Build auto-installer
kickstart, jumpstart, autoyast
installer does patches and runs cfengine after install

Patching

Most important

Patch early, patch often

Look before you leap

Copyright 2005 Abe Singer

Almost all breakins are the result of lack of patches

Apply patches as soon as possible

Strategy: install patches on test machine before on production machines
Test, let run for a little while, then install everywhere

Patch prioritization: is it a “critical” patch? Is it a security patch?
Is there an exploit?

Must balance risk of instaling patch with risk of intrusion

How to: Redhat, Up2Date or yum
Solaris: addpatch and patchk.pl

Cfengine Tutorial

Copyright 2005 Abe Singer

Cfe

Cfengine Concepts

Classes

Rules and Actions

Triggers

Copyright 2005 Abe Singer

Classes – each host belongs to one or more classes. What's done on the host is dependent on which class it's in

---

Some classes, such as type of OS, are assigned automagically

Others you assign by hand in the master config file

Rules are the things that are done for a given set of classes. There are lots of rules, but the key ones are the ones used to copy files, create directories, maintain perms, and execute a command when a config file has changed

---

Triggers are things that a rule creates that trigger actions to happen at a later time

Cfengine Rules

```
<rule>:
     class::
          arguments
```

Copyright 2005 Abe Singer

Cfengine rules consists of the name of the rool, followed by lines for each class to which things are done. Class can be omitted, in which case it applies to all of them, or the class “any” can be used

The class can even be the hostname of a single machine

Multiple classes can be put on a single line, separated with “|”. A particular host or class can be excluding by preceding it with a “!”

Cfengine Rules

Directories
Copyfiles
Links
Processes
Editfiles
Shellcommands

Copyright 2005 Abe Singer

These are the rules we're going to look at.

Directoires makes sure directories exist and have the right perms

Copyfiles compares files from the "reference" area to that on host, and copies them if they don't match, and sets permissions

Links is like directories, but for links

Processes allows you to send signals to processes after their config files have changed

Editfiles allows you to edit files in place making sure that they do or do not contain particular lines

Shellcommands allows you to run arbitrary shell commands

Revision Control

Track what changed and who

RCS

```
co –l <filename>
ci –u <filename>
rcsdiff <filename>
```

Copyright 2005 Abe Singer

Revision Control allows you to track changes in files, and who changes them, by recording a diff of each file

---

I use RCS. I uses the notion of “checkout” of a file.

Records data in the RCS directory in the local dir

“locks” files so that only one person can check it out at a time

Keeps a descriptive log of changes

The three essential commands:

---

checkout with lock

checkin with unlock

rcsdiff to see what changed

Building a Cfengine Reference

Start small

Have a "reference area"

Moves changes into ref

Copyright 2005 Abe Singer

Don't' try and do everything at once. Start maintaining a handful of common config files, such as inetd.conf and resolv.conf and hosts.allow

Create a "reference" area where these things are put. The layout of the reference area should match the hosts, e.d. /reference/common/root/etc/inetd.conf

Do host specific things by class, not by host. E.g. have a class called "webserver" for web servers

As you make changes for a host, copy them into the ref once you have them working.

Master config file

```
Groups:
  webservers = (web1 web2 web3)
  special = (`myshellscript`)
Import:
  webservers:: cf.webserver
Control:
  any::
     base_data = ( /refsys/Data )
     common = ("$(base_data)/common/root")
```

Copyright 2005 Abe Singer

The main config file for cfengine, cfagent.conf, is where you define classes for hosts, and which config files are loaded for each class

Groups is where you define what classes hosts belong to. Cfengine also automatically defines classes for OS type and revision, and some other things

Import specifies what other config files are read for a given class, this one way to control things that are only done for a certain class -- obviates need for lots of "!" in rules

Control allows you to define variable-ish things for use later, such as base directories. You can also define the order in which rules are processed and such

Recommend having a site-wide cf file, OS-specific, and per-service

Managing Config Files

Use "common" for common files

```
copy:
     any::
          $(common)/etc/inetd.conf
               d=/etc/inetd.conf o=0 g=0
               define=cmd_hup_inetd
```

Copyright 2005 Abe Singer

Config files are easiest. For configs that a common across all hosts, have a /ref/common/... directory.

Use OS class for OS-specific fhings

Use your own class for servers that you config, etc.

Installing Software

Just like config files

```
Copy:
  linux::
     $(archdir)/usr/local/bin/ssh
          d=/user/local/bin/ssh m=511 o=0 g=0
```

Copyright 2005 Abe Singer

Software is instlaled the same as config files

A good idea for important binaries, such as setuid programs

Pay close attention to perms of files created

Maintaining Permssions

```
Files:

  any::

    /etc/ssh_host_key m=0600 o=0 g=0 action=fixplain

    /etc/ssh_host_key.pub m=0644 o=0 g=0
  action=fixplain
```

Copyright 2005 Abe Singer

Running Cfagent

/etc/init.d/cfagent

Run nightly and at boot

Read output and look for errors

Copyright 2005 Abe Singer

Cfengine and Secrets

"too many secrets"

-- *Sneakers*

Copyright 2005 Abe Singer

You'll want to think hard about whether or not you should use cfengine to maintain "secret" information such as host keys and shadow files. Depending on how you use it, you may not be able to assure the confidentiality of the information (e.g. running it across NFS). Additionally, if you reference is compromised, so are all those secrets.

May be able to run cfagent against the CFD server with an ssl-wrapped connection, which would at least get you encryption on the wire.

Configuring Services Securely

Copyright 2005 Abe Singer

Secure Services

Isolation

Non-root

Copyright 2005 Abe Singer

Separate servers for each service. If you can't, try to jail the service

Chroot is easy to screw up, and there are ways to break it

Remember: there's no protection from the super-user

Don't run as root if at all possible, you can do it with more than you think, e.g. named

Make sure authentication is immune to passive attacks

Don't believe in things like rbash, they're too easy to break and misconfigure. Go with principle of least privilege

Time

Have a good time

It's not that hard

NTP

Copyright 2005 Abe Singer

Time: it's what's for dinner

File times in sync between hosts and file server, very important

Logging -- you want timestamps that are accurate across machines

accurate file modification times for forensic purposes

lack of time Can break some things like keberos and other time-based authentication

It's trivial to set up these days, lots of people get it by default now

---

NTP

Get time from various servers on the network, or your own receiver

Your server consults higher stratum servers, the rest of the machines on the network consult yours

And you can do it yourself for about $80, with a USB gps dongle. XNTP has all the hooks, and some versions are precompiled to support it. I'm doing it at home, it took longer to read the documentation than it did to actually get it working!

NTP Server

```
Server
  peer time.sdsc.edu
  server clock.isc.org
  restrict default notrust nomodify
restrict 132.249.0.0 mask
  255.255.0.0 nomodify

Sources
```

Copyright 2005 Abe Singer

Configure as client to some number of servers on the net. Restrict outside access to only those time servers

Default notrust nomodify

Inside: nomodify

Ntp is udp -- need to make sure you have anti-spoof filters

A list of sources can be found at ntp.org

Can also get a WWV receiver or GPS receiver (really cheap!) and configure your own stratum 1 server. It's really easy

Three or so servers should be enough

Clients chime off of your server(s).

Server <server>

Restrict to that server

Ntpdate on boot to get clock in synch -- ntpd won't synch if clock is too far off

NTP Client

Clients

```
peer ntp1.sdsc.edu
restrict default notrust nomodify
restrict 132.249.0.0 mask
  255.255.0.0 nomodify
```

Copyright 2005 Abe Singer

DNS

Zone transfers

Recursive Queries

Redundancy

Copyright 2005 Abe Singer

Consider authenticating zone transfers, although it can be interesting to see who is looking at your network

Require root access to change zone file

Make sure only one primary server

Have a remote secondary if possible

Make sure reverse matches forward

Check log file to make sure new zone loads -- serial number is often the culprit

NFS

No unprivileged mounts

Consistent UIDs

Export to explicit hosts

Root squashed, nosuid, nodev

Copyright 2005 Abe Singer

Don't allow high-port mounts
Make sure uids are consistent across clients, as everything is done by uid
Export to an explicit list of hosts
No default route, limit routes to "trusted" networks
Make sure root is squashed -- keeps attacker from jumping between hosts
Nosuid and nodev -- they have to be on local disk

And there are other network filesystems, e.g. OpenAFS. YMMV

SSH

No root login

Ssh keys? Tokens?

Copyright 2005 Abe Singer

Don't allow remote root login

Consider whether you want to accept ssh keys, etc. There are plusses and minuses.

Remember, if an attacker controls the remote endpoint, it doesn't matter if the user uses a password or an ssh key, the attacker has access to both.

token-based authentication helps, but the attacker can still hijack the session. If you're lucky, the user will notice this.

FTP

Jail

Anonymous only, or "group" accounts

Writable directories are not readable by ftp server

Copyright 2005 Abe Singer

HTTP

Use SSL for anything that authenticates

Kerberos module

Run as different user(s)

Manage CGI/PHP/whatever

Block known-bad stuff

Copyright 2005 Abe Singer

Be sure to use SSL for any pages that do authentication, whether it's browser-based authentication (e.g "basic" auth) or implemented via a form.

There's a kerberos module that does kerberos with clients that support it. Mozilla does if compiled with it (the RH distro is) and so does IE6, although I haven't figured out how to do a kinit on a windoze box to anything other than a domain controller/AD server.

The kerberos module is mod_auth_kerb, available from
http://modauthkerb.sourceforge.net/

It supports both real kerberos authentication, plus using basic auth and doing kerberos auth on the backend. Works okay!

Run each web server as a different user, and dont' have that user be a member of an groups that it doesn't need to be in. The website doesn't have to be owned by that user, and should not be writable by the user except in those areas where it *needs* to be. If you are running multiple servers, or hosting distinct sites, consider several instances of the server, each running as a different user, so that a compromise of one can't affect another.

And make sure your config files not writable by the http user!

and the place where you can still get screwed are broken CGI/PHP scripts. Remember that a CGI script is basically allowing an anonymous remote user to run programs on your machines.

If you're writing your own, there are various guides on writing secure apps. Primary thing is to verify any input from the client, make sure it's what you expect it to be, and doesn't have any characters in it that it should have. Wherever possible, map user input to a lookup table so that you don't actually use their input, it will also help you detect invalid input. It will make your apps more robust in the process!

SAMBA

Username mapping

Limit network access

No root user mapping

Copyright 2005 Abe Singer

We use a windows domain controller, samba is not PDC
Trusts that windows names are accurate, maps them to unix usernames
Usermap=<file> is used to map non-unix usernames
Limit access to server to only windows network
Security=domain
Encrypt passwords = yes
Invalid users = root

EMAIL

Postmaster and abuse

Root access to configs

Copyright 2005 Abe Singer

We're not talking about anti-spam measures, that's something entirely, and I dont' think if it as a security problem (yeah, yeah, viruses and worms -- stop using Outlook and you'll spend less money and time than all the other preventative measures combined.

Make sure you accept mail to postmaster and abuse, and that someone actually *reads* it. RFC actually requires that you accept mail for postmaster. Mail sent to these (other than the spam that they get) will sometimes alert you to a problem on your network.

Make sure you have to be root to change mail config files, including aliases.

Shared apps

/usr/apps

read-only

/usr/local/bin/whatever -->
/usr/apps/whaever-1.0/bin/whatever

Copyright 2005 Abe Singer

How we manages shared application packages, anything that we build ourselves, such as pgp, servers, php, compilers, whatever

We have a directory where all packages are installed, each under their package name. For example, /usr/apps. It could be anything you want

The directory is mounted *read-only* from client machines. So a compromise of a client machine can't be used to trojan applications that are used globally, esp. servers. Different architectures mount different directories

An "install" host has write permission to the directory. Token-based auth controls acces to that host. One install host per architecture.

Packages are built using "/usr/apps" as the install prefix. We encourage using a shell scripts which untars the source tarball in scratch space, does all the building, with switch statements for each architectures, and then does the install. That way you don't have to do so much when upgrading to a new version, it's just a matter of changing a line or two on the build script.

And, as necessary, things in /usr/local/bin are symlinked to /usr/apps, so that people don't have to maintain all kinds of things in their path

This also allows you to keep around different versions, people who need an older version can reference it explicitly in /usr/apps, people who just want the current version can use /usr/local/bin.

Other Stuff

Copyright 2005 Abe Singer

Auditing hosts

setuid/setgid files, esp. root

world-writable files and dirs

root path

daemons

user accounts

root account

Copyright 2005 Abe Singer

Ever been told to go audit a host for security? Wondered where to start?

A relatively simple sets of tasks for auditing a host for security. These are also a good set of things to follow for building a reference system

first, get a list of all files on the host that have the setuid or setgid bit set,

---

especially those owned by root. Which of them do you actually use? Of those, do they *need* to be setuid? Many don't! And if you don't know what they are, find out!

Next, look at any files and directories that are world writable.

Directories with the sticky bit set are usually okay, but for anything else, there's almost *never* a reason for the file to be world writable.

Check all the directories and files in root's path

---

and make sure that they're owned by root and not writable by anyone else. This includes parent directories. And no "." in root's path

What daemons are running,

---

especially those listening to the network, or to a Unix domain socket (lsof is your friend here)

include (x)inetd here. How are they configured? Do they have appropriate access control? Encryption and authentication where necessary?

Check user accounts, remove any role accounts that aren't necessary.

Evaluating Software

Authentication

Access Control

Privileged access

Copyright 2005 Abe Singer

How to evaluate software for use in your infrastructure

Does it listen on the network?

Does it require authentication? How? Users accounts or other? Plaintext or strong authentication? Can it integrate with your infrastructure (e.g. Kerberos)

Does it run setuid or setgid? How does it control access to files and such if so? Does it create files or directories? With what permissions?

Does it have to run as root? Really? Can it be jailed?

Does it require a role account? What does the account need access to?

What does it install where? Does it try and install accounts or groups

Does ti spawn shells? Under client control?

Does it log to syslog or other?

Consider profiling a testrun, see what it accesses, what ports, etc.

Does it properly validate input?

If you don't like it, is there an alternative?

OR just say "no"

Centralized Logging

Diagnostics

Utilization

IDS and Forensics

***.debug @loghost**

Copyright 2005 Abe Singer

Logging is important for many reasons, see other pesentation

Forward everything to a centralized loghost

Burglar Alarms

Sudo and su

test.cgi

.cfsaved

credentials

Copyright 2005 Abe Singer

Simple things you can do to detect *some* malfeasance

Make sure sudo and su log and email on failures

Test.cgi, linked to test, phf, etc. is lots of fun, esp when somebody runs it from a tty. Also other known exploitable cgi scripts (g*d are there a lot of them!) xmlrpc.php is popular these days.

Check for .cfsaved files, esp. on setuid programs

User credential changes, esp. certs, ssh keys, accounts being created by something other than acct. management.

Interesting user logins from places you've never seen before, users logging in from hosts who have ssh-scanned you or portscanned you are interesting.

hide links on your web pages, look for hits on them in your logs, will pick up someone mirroring your site, scraping, etc. Of course, you'll want to whitelist Google et. al.

Integrating Windoze

Ghost

SMS

AD

Network Filters

Copyright 2005 Abe Singer

We're not severely well integrated between windows and unix

Shared printing via windows print server, could be done with cups instead

Login and home directories not shared, but could be with samba
Windoze users have their unix dirs available via samba

Windows ref: winddows install tweaked, then ghost image created and loaded from ghost server
---
Have to maintain about a dozen ghost images due to hardware differences

Accounting done through active directory domain controller

Patches:
Norton NAV Corporate 9
SMS for patching, management
HRNetCHeck Pro for things that SMS doesn't address; good for auditing and reporting
---
Accounting done through active directory domain controller

XP ref build: mostly lock down IIS (IE?)
IESpyAd -- tool from UI that creates big list of untrusted sites

Default users get write privs to local disk -- they have to
Domain group policy overrides where necessary

Network filters for windows file sharing, remote desktop

Where firewalls fit in

Machines which can't protect themselves

Copyright 2005 Abe Singer

Some say that firewalls are for machines which can't protect themselves

Like printers which have web servers embedded, although we use RFC1918 for that

Can be useful for detection and logging of certain activity, or to block known bad activity, esp. during incident response

Although you can do a lot of this with tcp-wrappers and other host-based stuff

Critical Systems

Limit access

Two-factor authentication

Watch closely

Copyright 2005 Abe Singer

Think carefully about critical systems

Who has access, and should they? Limit account to only those who need 'em.

Even on the file server, you can have data owned by users who don't have an account on that host

Consider using two-factor authentication, such as token-based, for access

---

And watch activity on them closely. Who is logging in? What did they do? Are there things running that shouldn't be?

SAR or SA is your friend

Resources

1 person per OS

Copyright 2005 Abe Singer

The resources required to maintain the environment

We have one person per OS to maintain the Ref for that OS. They also maintain infrastructure services

That's in addition to the people who physically install them

Response plan

who gets called?

who makes decisions?

who's responsible for which machines?

Where are the hosts located?

Copyright 2005 Abe Singer

Sh*t is gonna happen, no matter what, so have a response plan

Make it clear who gets notified of possible security problems. First person to call is the security responder, who should be responisble for notifying anyone else. Good idea to notify management when the call comes in, and get back to them after investigating. They don't like surprises.

Make sure it's also very clear who gets to make the decisions abouut what to do. Who decides to take a host offline, and when? You can get pressure from different directions, some people want to keep services up, not annoy users, etc. You can't end up in a standoff where a machine should be taken offline and someone is refusing to do it, and doesn't recognize your authority.

Have a list of who's responsible for machines or service, including their home phone numbers and cell numbers, etc. Some will be reticent to give it, just point out to them that if you can't reach them and their system has a problem, you'll just have to unplug it and they get to clean up the mess.

And be able to find machines quickly. This sounds simple, but can be difficult in a big machine room, or when hosts are stuck in closets, or under people's desks!

Responding

Don't panic

Stay off the machines

Gather information

Plan, then act

Report

Copyright 2005 Abe Singer

A few tips on how to reponde

First, don't panic. intruders may be in, they may not be. 90% of the time it's a false alarm. And you don't want to make things worse

Stay off the hosts in question, if you know which they are, and keep everyone else off too. When there's an incident, everyone wants to start logging in and see what's going on, which can tip off an intruder. Keep in mind that your intruder may be watching you as much as you're watching him/her/them/it.

First, gather information. How was it discovered, what information is there already. It's up to you to confirm that there's actually an incident. sometimes people misinterpret what they're seeing, and tell you their interpretations instead of the the facts. And be careful about gathering. If you can, don't log onto hosts, use something non-interactive, such as rsh or ssh (in rsh mode).

You want to both verify the intrusion, and that you know the scope of it. Cleaning up and missing a backdoor is just going to make you look bad and lose sleep.

Once you're sure what's going on, make a plan of action, and then execute it. Delegate where you can, but verify that measures are being done appropriately, and in the right order.

Then report out what you did, including followup things that should be done after the fact. Sending out the report before your managers wake up usually scores points!

The End

Copyright 2005 Abe Singer